Area–Delay–Energy Efficient VLSI Architecture for Scalable In-Place Computation of FFT on Real Data

Basant K. Mohanty, *Senior Member, IEEE*, and Pramod Kumar Meher, *Senior Member, IEEE*

***Abstract*— Efficient computation of real-valued fast Fourier transform (RFFT) has received significant attention in recent years due to its several applications in conventional digital signal processing and other emerging areas. In-place RFFT architectures are gaining popularity due to their lower hardware complexity compared with pipeline architectures. But the scaling of in-place RFFT architecture for higher lengths and higher throughput is a challenging issue due to increasing memory access conflict and higher memory bandwidth requirement. In this paper, a design approach is presented to develop an area-delay and energy-efficient architecture for in-place RFFT. Generally, an in-place fast Fourier transform (FFT) structure consists of a *butterfly block* which performs a set of butterfly operations in every clock cycle. From complexity analysis we find that in-place FFT structures with larger butterfly blocks are more efficient in terms of area-time complexity and energy consumption. The resolution of memory access conflict is however more challenging for higher butterfly block sizes. Therefore, we have analyzed the data-flow and memory footprint of in-place RFFT architectures for different throughput requirements, and based on that, we have proposed here a strategy to partition the storage unit into several banks of smaller sizes (without increasing the overall memory size) to resolve the memory access conflicts by concurrent data-swapping between the banks. Synthesis result shows that the proposed structure with butterfly block of size 4 and 8 involves (∼44% and ∼57%) less area-delay product and (∼54% and ∼57%) less energy per sample than those of existing similar structure on average for different FFT lengths, respectively.**

***Index Terms*— Fast Fourier transform, in-place computation, real-valued FFT.**

I. Introduction

FAST Fourier transform (FFT) is widely used in several applications such as spectral estimation, speech and audio processing, image processing, digital communication, wireless sensor network (WSN), radar signal processing, bio-medical signal processing and many more. Due to increasing demand for real-time and high data-rate multimedia services, the data-rate of wireless communication systems has been steadily increasing. In these applications, FFT computation is performed on sequences of wide range of lengths and sampling rates. FFT size vary from 64 to 32768 and the sampling rate generally vary from a few Mega-samples/s to Giga-samples/s in different applications. It is therefore, necessary to have computation of FFT of large sizes with adequately high-throughput according to the requirement of the applications.

Realization of FFT of larger sizes in resource constrained environment and providing high-throughput rate is a challenging task. FFT architectures are broadly of two contrasting categories: pipeline architecture (*i.e.,* single-path delay feedback (SDF) based structures [1] and multi-path delay commutator (MDC) based structures [2], [4]–[6]) and processor-memory (PM)-based folded in-place structures [9]. In general, all these structures consist of three main functional units namely, the: (i) butterfly computation unit (BCU), (ii) storage unit (SU), and (iii) twiddle factor generation unit (TFU). The MDC-based structures are mostly preferred for pipeline computation of FFT due to their regular data-flow and higher utilization efficiency of BCU at the cost of marginally higher memory requirement than the SDF-based structures. In the PM-based structures, intra and/or inter butterfly stages of computations are folded in the BCU and take advantage of in-place FFT for low-complexity realization. Pipeline and PM-based in-place architectures are two main variants of FFT structures with contrasting features and the choice of one of these two categories of architectures largely depends on the space-time constraints of the target application. Therefore, several pipeline and PM-based architectures have been proposed for efficient implementation of FFT for different applications.

There are several applications such as speech, audio, image, and video processing, where FFT of real-valued signals is used [3]. Efficient realization of FFT of real-valued signals has received further attention now-a-days due to the emergence of biomedical signal processing and wide applications of real-valued time-series analysis. Some efforts also have been made to develop specific architectures for real-valued FFT (RFFT). The data-flow of RFFT algorithm becomes increasingly disordered for higher FFT sizes and that makes the in-place RFFT design very challenging due to increasing memory access conflict [9]. Some PM-based in-place architectures have been proposed for RFFT using specialized packing algorithms [7], [8]. An in-place architecture and conflict-free memory addressing scheme for RAM-based memory banks have been proposed for continuous processing of RFFT [10], [11]. The SU of N-point in-place RFFT architecture is implemented using $2N$ single-port RAM words. Recently, a register-based SU design is presented in [12] for in-place RFFT architecture to save the memory footprint. It is observed that a pair of 2-point butterflies (referred to as *butterfly block* of size 2) is used in the existing RFFT designs.

Manuscript received June 4, 2018; revised August 19, 2018; accepted September 14, 2018 . This paper was recommended by Associate Editor Y. Yu. *(Corresponding author: Basant K. Mohanty.)*

B. K. Mohanty is with the Department of Electronics and Tele-Communication Engineering, SVKM'S NMIMS, Shirpur Campus, Shirpur 425405, India (e-mail: bk.mohanti@gmail.com).

P. K. Meher is with the C. V. Raman College of Engineering, Bhubaneswar 752054, India (e-mail: pkmeher@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2018.2873720

Due to in-place nature of the computation, the SU complexity increases linearly only with FFT length and contributes significantly to the total area for lengths higher than 32. During any particular clock cycle, the RFFT structure of butterfly-block size 2 utilizes only 4 memory words out of N words stored in the SU. This results in low utilization of SU. For small butterfly blocks *the combinational logic is a small fraction of the storage area* and thus the storage is heavily under-utilized. On the other hand if we increase the butterfly block size, then with a small increase in overall hardware complexity, not only the memory can be better utilized but also the throughput, area-delay product, and power-performance can be improved substantially. The scaling of in-place RFFT architecture for higher lengths and higher butterfly block sizes, however, is a challenging issue due to increasing memory access conflict and large memory bandwidth requirement.

Conventional storage structures and SU design does not support higher butterfly block size due to memory access conflict. We find that the SU could be partitioned into certain number of banks and suitable scheme for swapping of stored data across those banks could be developed to avoid memory access conflict resulting due to larger butterfly blocks. We have analyzed the SU complexity along with the memory footprint for different throughput to study the memory access per output and to identify the design constraint for conflict-free memory access. Based on that, a design approach is proposed here to develop area-delay efficient architectures for in-place RFFT computation. The rest of the paper is organized as follows: Complexity analysis of in-place RFFT and the proposed design strategy is presented in Section II. Proposed architecture for RFFT is presented in Section III. Hardware and time complexities are discussed in Section IV. Finally conclusions are presented in Section V.

II. Complexity Analysis and Design Strategy

In this Section, we analyze the complexity of in-place RFFT architecture for different throughput rates and FFT lengths to arrive at the proposed design strategy. The BCU of $(L/2)$ butterfly block size involves L real-multipliers and $(3L/2)$ real-adders.[1] The SU can be implemented using RAM or registers. However, we use a register-based design for the SU for simultaneous read/write operations during the in-place computation of N-point RFFT. Therefore, the in-place RFFT structure involves L real multipliers, $(3L/2)$ real-adders, N registers and computes N-point RFFT in $\{(N/L)\log_2 N\}$ clock cycles. Interestingly, the complexity of BCU is independent of FFT size and depends on the butterfly-block size while the complexity of data-storage unit is independent of butterfly-block size and depends on the FFT size. The computation time which increases with the FFT size could be reduced by increasing the butterfly block size according to the throughput requirement.

The hardware complexity of the existing register-based in-place architecture is estimated in terms of multipliers, adders, registers and computation time in clock cycles for different butterfly-block sizes and FFT sizes. The estimated values are listed in Table I. The memory utilization efficiency[2] is also estimated for different butterfly-block sizes and FFT sizes and shown in Table I. As shown in Table I, for large size FFTs, memory complexity is significantly large compared to multiplier and adder complexity of the structure having small butterfly-block sizes. The in-place structure has low memory utilization efficiency for small butterfly block sizes and large FFT sizes. Interestingly, the memory complexity is independent of butterfly block sizes. Consequently, the memory utilization efficiency increases for higher butterfly block sizes which is maximum for $L = N$ (full-parallel design). Therefore, the in-place FFT structures have the potential to be more area-delay efficient for higher butterfly block sizes.

TABLE I
Hardware and Time Complexities of In-Place RFFT Structure

FFT size N	BFB size $(L/2)$	MULT (L)	ADD $(3L/2)$	REG (N)	CT (ccs)	MUE (%)
32	2	4	6	32	40	12.5
	4	8	12	32	20	25
	8	16	24	32	10	50
	16	32	48	32	5	100
64	2	4	6	64	96	6.25
	4	8	12	64	48	12.5
	8	16	24	64	24	25
	16	32	48	64	12	50
128	2	4	6	128	224	3.125
	4	8	12	128	112	6.25
	8	16	24	128	56	12.5
	16	32	48	128	28	25
256	2	4	6	256	512	1.56
	4	8	12	256	256	3.12
	8	16	24	256	128	6.25
	16	32	48	256	64	12.5

LEGEND: BFB: butterfly block, MULT: multiplier, ADD: adder, REG: register, CT: computation time = $(N/L)\log_2 N$, UT: utilization efficiency = $(L/N)\times 100$, ccs: clock cycles

We have used CMOS transistor count to estimate the area complexity, where radix-4 Booth multiplier of [13] is used to implement the multiplier, carry propagate adder (CPA) to implement adder and D-flip-flops to implement register. The multipliers, adders and registers required by different designs listed in Table I are implemented using standard cells {2-input XOR/XNOR, 2-input AND, 2-input OR, NOT gate, 2:1 MUX, half-adder, (HA), full-adder (FA) and D-FF}. Area complexity is estimated using the formula:

$$MUL_a = 12N_x + 6N_a + 6N_o + 2N_n + 12N_m + 18.N_h + 28N_f \quad (1a)$$

$$CPA_a = 12N_x + 18.N_h + 28N_f \quad (1b)$$

$$REG_a = 24N_d \quad (1c)$$

where, N_x, N_a, N_o, N_n, N_m, N_h, N_f and N_d represents standard cell counts of {2-input XOR/XNOR, 2-input AND, 2-input OR, NOT gate, 2:1 MUX, half-adder, (HA) and

[1]The multipliers and adders of real-valued data are, respectively, refereed to as real-multipliers and real-adders in this paper.

[2]memory utilization efficiency = (number of memory words read per clock cycle (L) / memory complexity (N))$\times$ 100

TABLE II
AREA AND TIME COMPLEXITIES OF IN-PLACE RFFT STRUCTURE IN TERMS OF CMOS TRANSISTOR COUNT AND COMPUTATION TIME

N	$L/2$	**Storage unit area** $\times 10^3$			**BCU area** $\times 10^3$			**Total Area** $\times 10^3$			**ADP**$\times 10^5$			**ADP Saving** (%)		
		8	12	16	8	12	16	8	12	16	8	12	16	8	12	16
	4				18.8	40.2	69.5	21.2	44.8	75.6	1.8	3.6	6.1	12	9	8
	8				37.7	80.3	139.0	40.8	84.9	145.1	1.6	3.4	5.8	18	14	11
32	2	6.1	9.2	12.3	9.4	20.1	34.7	15.6	29.3	47.0	6.2	11.7	18.8	−−	−−	−−
	4				18.8	40.2	69.5	25.0	49.4	81.8	5	9.8	16	20	16	13
	8				37.7	80.3	139.0	43.8	89.6	151.3	4.5	9	15	30	24	20
	16				75.4	160.7	278.0	81.5	170	290.2	4	8.5	15	35	28	23
64	2	12.3	18.4	24.6	9.4	20.1	34.7	21.7	38.5	59.3	21	37	57	−−	−−	−−
	4				18.8	40.2	69.5	31.1	58.6	94.1	15	28	45	28	24	21
	8				37.7	80.3	139.0	50.0	98.8	163.6	12	24	39	42	36	31
	16				75.4	160.7	278.0	87.6	179.1	302.6	11	22	36	50	42	36
128	2	24.6	36.9	49.2	9.4	20.1	34.7	34.0	56.9	83.9	76	128	188	−−	−−	−−
	4				18.8	40.2	69.5	43.4	77.0	118.6	49	86	133	36	32	29
	8				37.7	80.3	139.0	62.3	117.2	188.1	35	66	105	54	49	44
	16				75.4	160.7	278.0	100.0	197.5	327.1	28	55	92	63	57	51
256	2	49.2	73.7	98.3	9.4	20.1	34.7	58.6	93.8	133.0	300	480	681	−−	−−	−−
	4				18.8	40.2	69.5	68.0	113.9	167.8	174	292	430	42	39	37
	8				37.7	80.3	139.0	86.8	156.1	237.3	111	197	304	63	59	55
	16				75.4	160.7	278.0	124.5	234.4	376.3	797	150	241	73	69	65

LEGEND: BCU: Butterfly unit, DSU: data storage unit, ADP: area-delay product, ADPS: area-delay product saving, ADP = Tot. Area $\times$ computation time (in ccs), ADPS= [(ADP$_r$ - ADP)/ADP$_r$]$\times$ 100, where ADP$_r$ represents the ADP of reference design of butterfly block 2.

full-adder (FA), D-FF}, respectively and the multiplying constants {12, 6, 6, 2, 12, 18, 28, and 24} represent the CMOS transistor counts of standard cells {2-input XOR, 2-input AND, 2-input OR and NOT}, respectively, according to the TSMC 65nm GPLUS standard cell library data sheet. The area complexities of {multiplier (MUL_a), adder (CPA_a), and register (REG_a)} for bit-width $w = 8$, 12, 16 are estimated according to (1a) as $\{2034, 214, 192\}$, $\{4532, 326, 288\}$, $\{8030, 438, 384\}$, respectively, in terms of CMOS transistors count. Using the complexity estimates given in Table I the area complexity of butterfly unit (BCU) and storage-unit (SU) are also calculated in terms of transistor count. The estimated area-delay product (ADP) of all the designs are listed in Table II for discussion.

Multiplier complexity increases nearly 4 times when bit-width doubles while adder and register complexity increases proportionately. Consequently, the BCU and SU complexity increases by (2.13 times and 3.69 times) and (1.5 times 2 times), respectively, for bit-width 12 and 16 than those of bit-width 8. The SU complexity is independent of butterfly block size and changes proportionately with the FFT size while the BCU complexity changes proportionately with butterfly block size and independent of FFT size. The area complexity of the overall FFT structure does not change proportionately with the throughput requirement (and the butterfly block size). Further, relative increase in total area with the butterfly block sizes marginally increase with change in bit-width. Therefore, the use of larger butterfly block in the in-place FFT design could be more area-delay efficient and energy efficient. Also, it can be observed from Table II that for a given FFT size, the in-place FFT structure offers more saving of ADP when the butterfly block size doubles. We find that when the SU and BCU complexity are relatively close to each other for a given FFT size, the use of higher butterfly block size can offer relatively higher ADP saving than other cases. This complexity analysis reveals that larger butterfly block sizes (4 and 8) should be used to develop in-place FFT structures for FFT lengths more than 16 and 32. However, butterfly block sizes 16 could be used for FFT sizes higher than 256 to get higher ADP saving.

We observe that when butterfly block-size doubles, each memory bank is split into two parts which form two separate banks with reduced size. However, the bank memory-address need not be split into two separate sets. Consequently, when the memory banks are split into two separate banks then those two banks share a common set of memory addresses. Therefore, the addressing mechanism of memory-banks could be simplified when the number of banks increases. Due to reduction in bank size, the memory access delay is also reduced. This is an interesting feature which could be exploited in the proposed register-based storage design.

The SU comprises of L banks. Each butterfly stage of the FFT flow-graph produces N intermediate outputs and these intermediate outputs are consumed immediately by the next butterfly stage. However, a butterfly stage receives the intermediate outputs in an order different from the order in which they are produced. Bank conflict arises due to the necessary reordering of data during the computation of subsequent different stages. Accordingly, data swapping between a pair of memory-banks is performed before and after the memory-write and memory-read operations to resolve the access conflict. As discussed in [12], data swapping between two pairs of banks are performed to resolve the bank conflict of in-place DIT-FFT for butterfly block-size 2. The number of banks are doubled when the butterfly block-size is doubled. For example, the number of banks is 8 and 16, respectively, for butterfly block-size 4 and 8. The memory access conflict arises in 8 banks for butterfly block-size 4 and in case of butterfly block-size 8, it arises in 16 banks. To resolve these conflicts, data swapping need to be performed between 4 pairs of banks for butterfly block-size 4, and in case of butterfly block-size 8, data swapping need to be performed between 8 pairs of banks. Therefore, more data-swapping between the banks need to be performed to resolve the conflicts when block size increases. The design of suitable data-swapping circuits and memory banks to support the desired bandwidth are important to develop efficient hardware structures for in-place RFFT computation with higher block-sizes. To design the data-swapping unit detail, the data-flow analysis of storage unit of RFFT computation for a given butterfly block-size for different butterfly stages need to be performed to identify all the bank conflict instances and to note the data-swapping between the banks. Based on these observations, we present here a design approach to develop the proposed architecture for in-place RFFT.

1) The combinational logic complexity and I/O pins increases proportionately with the butterfly block-size; and the SU complexity is independent of butterfly block-size. According to the available resources and I/O pins, butterfly block size should be chosen suitably to design in-place RFFT structures.
2) A data-flow analysis of memory unit for the given block-size need to be under taken to identify the instances of memory access conflicts which could be used for the design of memory structure and data-swapping unit in order to resolve the access conflicts.
3) A multi-channel register-based memory bank structure need to be designed to achieve the desired feasibility of multiple read and write in a given clock cycle.

Following the aforementioned approach, in the next Section, we present the proposed register-based storage unit for in-place DIT-RFFT with butterfly block-sizes 4 and 8. A parallel structure of in-place DIT-RFFT is then derived using the proposed storage unit and the data swapping unit.

III. Proposed Architecture

The SU consists of L register-banks of depth P words each for butterfly block size $(L/2)$, where $P = N/L$, L is the input block size and N is the FFT size. During every clock cycle, a block of L data is read from L banks for parallel computation of $(L/2)$ 2-point butterfly, and stores one block of L outputs back into the same banks. The reordering of intermediate data has no effect when the entire computation of a butterfly stage is performed in one clock cycle (referred to as *full-parallel* structure). Consequently, the SU of the full-parallel structure is comprised of N banks where each bank stores only one sample. When butterfly computations of a particular butterfly stage are folded and performed in different clock cycles then the SU of folded structure need to be split into L banks where each bank stores $(P = N/L)$ intermediate values. We can find that $(P = 2^f)$, where f is the folding factor. The reordering of intermediate data could result in access conflict when a bank stores more than one intermediate data. It is observed that the number of reordering of data increases from the first stage to the final butterfly stages. The bank conflict associated with the butterfly computation of different stages depends on the folding factor (f). The bank conflict starts from $\{S - (f - 1)\}$-th butterfly stage to $\{S\}$-th butterfly stage, where $S = \log_2 N$. Therefore, the access conflict arises early in the butterfly stages for higher folding factor and late in the butterfly stages for smaller folding factor. To avoid these access conflicts, data-swapping between the banks is performed before writing data into the banks. Moreover, it is observed that the data read from the SU are not in the order as required by a particular butterfly stage and they need to be reordered. Therefore, data-swapping need to be performed before and after the memory operations on the banks. Input-output data-flow analysis of the SU for different butterfly stages of RFFT need to be performed to identify the necessary data-swapping instances for resolving bank access conflict.

A. Data-Flow Analysis of Storage Unit

The BCU for butterfly block size $(L/2)$ takes (N/L) clock cycles to complete the computations of one butterfly stage and takes $(N/L)\log_2 N$ clock cycles to complete all the butterfly operations for N-point FFT. We have taken butterfly block size $(L/2) = 4$ and FFT size $N = 32$ as an example to analyze the input-output data-flow resulting in memory access conflicts and necessary data-swapping instances. The input-output data-flow of other butterfly block sizes can also be undertaken in similar manner. Computations of each butterfly stages of 32-point FFT is performed in a set of 4 successive clock cycles and the entire computation is performed in 20 clock cycles. During the first 16 clock cycles of a period of 20 clock cycles, the BCU performs butterfly operations to produce an 8-point intermediate output-data-block which is written back into the SU. During the last 4 clock cycles of a period of 20 clock cycles the FFT components are computed out as output of the BCU while samples of the next input sequence are stored in the SU during the same period.

The SU pertaining to butterfly block size 4 is comprised of 8 memory banks (B_1, B_2, B_3, B_4, B_5, B_6, B_7, and B_8), so that a block of 8 words can be retrieved/stored from/in the 8 banks in each clock cycle using a common address for reference. The output data-flow from/into the SU for the computations of different stages of two successive 32-point input vectors $\mathbf{x}_1 = \{x(0), x(1), \cdots, x(31)\}$ and $\mathbf{x}_2 = \{x(32), x(17), \cdots, x(63)\}$ is given in Table III for 20 clock cycles.

During the clock cycles $CC_{1.1}$ to $CC_{1.20}$ the FFT of input-vector $\mathbf{x}_1$ is computed. The input-vector $\mathbf{x}_1$ is accessed from the register-banks during the clock cycles $CC_{1.1}$ to $CC_{1.4}$ and the 32-point intermediate data-vectors $\mathbf{r}_1$ corresponding to butterfly stage-1 is loaded into the SU. During clock cycles $CC_{1.5}$ to $CC_{1.8}$, $CC_{1.9}$ to $CC_{1.12}$, and $CC_{1.13}$ to $CC_{1.16}$, intermediate data-vectors $\mathbf{r}_1$, $\mathbf{u}_1$ and $\mathbf{v}_1$ are accessed from the storage unit for computation of stage-2, stage-3, and stage-4 butterfly computations of 32-point RFFT. During $CC_{1.17}$ to $CC_{1.20}$, the intermediate data-vector $\mathbf{w}_1$ is accessed from the storage unit to perform the butterfly operation of

TABLE III
OUTPUT DATA-FLOW FROM THE STORAGE-UNIT FOR DIFFERENT BUTTERFLY STAGES OF IN-PLACE RFFT FOR BUTTERFLY BLOCK SIZE $L = 4$ AND FFT SIZE $N = 32$

sig	CC	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
		signal sample index							
$\mathbf{x}_1$ stage-1	$CC_{1.1}$	0	24	16	8	4	28	20	12
	$CC_{1.2}$	2	26	18	10	6	30	22	14
	$CC_{1.3}$	1	25	17	9	5	29	21	13
	$CC_{1.4}$	3	27	19	11	7	31	23	15
$\mathbf{r}_1$ stage-2	$CC_{1.5}$	0	16	8	24	4	20	12	28
	$CC_{1.6}$	2	18	10	26	6	22	14	30
	$CC_{1.7}$	1	12	9	25	8	21	13	29
	$CC_{1.8}$	3	19	11	27	7	23	15	31
$\mathbf{u}_1$ stage-3	$CC_{1.9}$	0	8	4	12	16	24	20	28
	$CC_{1.10}$	2	10	6	14	18	26	22	30
	$CC_{1.11}$	1	9	5	13	17	25	21	29
	$CC_{1.12}$	3	11	7	15	19	27	23	31
$\mathbf{v}_1$ stage-4	$CC_{1.13}$	0	4	2	6	16	24	18	26
	$CC_{1.14}$	8	12	10	14	20	28	22	30
	$CC_{1.15}$	1	5	3	7	17	25	19	27
	$CC_{1.16}$	9	13	11	15	21	29	23	31
$\mathbf{w}_1$ stage-5	$CC_{1.17}$	0	2	1	3	16	24	17	25
	$CC_{1.18}$	8	12	9	13	20	28	21	29
	$CC_{1.19}$	4	6	5	7	18	26	19	27
	$CC_{1.20}$	10	14	11	15	22	30	23	31
$\mathbf{x}_2$ stage-1	$CC_{2.1}$	32	56	48	40	36	60	52	44
	$CC_{2.2}$	34	58	50	42	38	62	54	46
	$CC_{2.3}$	33	57	49	41	37	61	53	45
	$CC_{2.4}$	35	59	51	43	39	63	55	47

stage-5 and deliver the output corresponding to the FFT of input sequence $\mathbf{x}_1$. During the same period, the next input sequence $\mathbf{x}_2$ is loaded into the storage unit. For in-place FFT computation, the intermediate output values are stored exactly in the same locations from which the corresponding input data were read. Therefore, the intermediate output vectors $\mathbf{r}_1$, $\mathbf{u}_1$, $\mathbf{v}_1$, and $\mathbf{w}_1$ are stored exactly in the same memory locations of input signals $\mathbf{x}_1$, $\mathbf{r}_1$, $\mathbf{u}_1$, and $\mathbf{v}_1$, respectively. One can find from Table III that samples of each input data-block of stage-2 and stage-3 computation are sourced from 8 different banks and no conflict arises while reading input data-blocks from different banks. However, during stage-4 of butterfly computation, 8 samples of each input-block are read from 4 banks. For example: sample of first input-block $\{v_1(0), v_1(4), v_1(2), v_1(6), v_1(16), v_1(24), v_1(18), v_1(26)\}$ are read from register-banks, (B_1, B_3, B_5, B_7) while samples of the second input-block $\{v_1(8), v_1(12), v_1(10), v_1(14), v_1(20), v_1(28), v_1(22), v_1(30)\}$ are read from register-banks (B_2, B_4, B_6, and B_8). Similar situation also arises for other input blocks of stage-4 as well as stage-5 computations. When more than one data are required from a particular bank during a clock cycle then an access conflict occurs. The access conflict can be resolved by swapping samples of each data-block in pairs before/after they are stored/read in/from the register-banks simultaneously.

The storage unit uses two data-swapping circuits (DS_1 and DS_2) to perform the required data-swapping on the intermediate data-blocks produced by the BCU to resolve the bank conflicts and reorder the samples of input-blocks

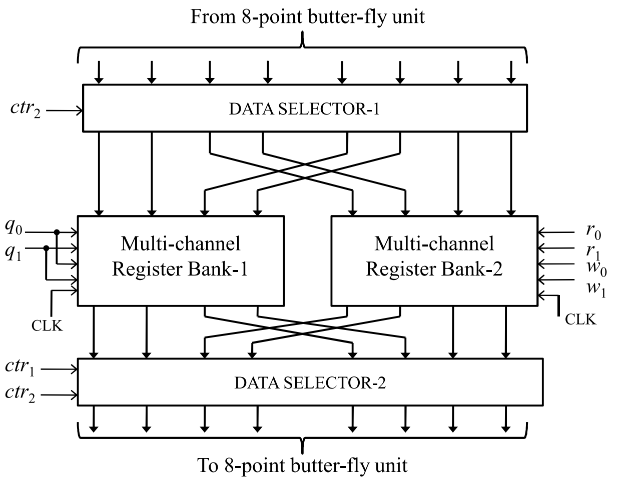
From 8-point butter-fly unit
ctr_2
DATA SELECTOR-1
q_0
q_1
Multi-channel Register Bank-1
Multi-channel Register Bank-2
r_0
r_1
w_0
w_1
CLK
CLK
ctr_1
ctr_2
DATA SELECTOR-2
To 8-point butter-fly unit

Fig. 1. Storage unit design of proposed RFFT structure for butterfly block-size 4 and $N = 32$.

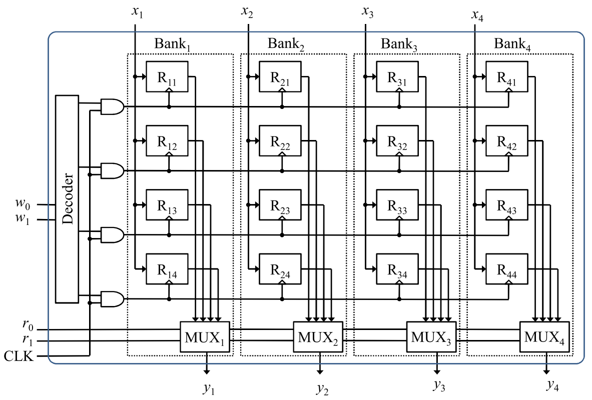
x_1
x_2
x_3
x_4
Bank$_1$
Bank$_2$
Bank$_3$
Bank$_4$
R$_{11}$ R$_{21}$ R$_{31}$ R$_{41}$
R$_{12}$ R$_{22}$ R$_{32}$ R$_{42}$
R$_{13}$ R$_{23}$ R$_{33}$ R$_{43}$
R$_{14}$ R$_{24}$ R$_{34}$ R$_{44}$
Decoder
w_0
w_1
r_0
r_1
CLK
MUX$_1$ MUX$_2$ MUX$_3$ MUX$_4$
y_1 y_2 y_3 y_4

Fig. 2. Internal structure of multi-channel register bank for butterfly block-size 4 and $N = 32$.

retrieved from the register-banks. The input-output data-flow of DS_1 and DS_2 is shown in Table IV and Table V for butterfly block size 4 and FFT size 32.

B. Storage Unit Design

The proposed structure for N-point in-place DIT RFFT computation is similar to the structure of [12] at the block level. However, the design of SU as well as the BCU are different from those of [12]. The design of SU of the proposed structure for 32-point RFFT and butterfly block-size 4 is shown in Fig.1. It consists of two multi-channel register-banks (MCRBs) and two data-selectors. The MCRB implements multiple memory banks in one unit which uses a common address decoding logic for all the banks. The MCRB-1 implements register-banks (B_1, B_2, B_5, B_6) while MCRB-2 implements register-banks (B_3, B_4, B_7, B_8). The internal structure of MCRB for four bank channels of depth 4 each is shown in Fig.2. It consists of 16 registers which are arranged in 4 rows and 4 columns. The four registers of each column form one bank channel, such that registers (R_{11}, R_{12}, R_{13}, R_{14}) form Bank-1 and registers (R_{41}, R_{42}, R_{43}, R_{44}) form Bank-4. A 2:4 decoder is used to decode the 2-bit write-address (w_0, w_1) and enable the clock signal CLK for a particular row of registers belonging to four different banks to write one block of input-data $\{x_1, x_2, x_3, x_4\}$. The content of all the four

TABLE IV
INPUT-OUTPUT DATA-FLOW OF DATA-SELECTOR-1 OF PROPOSED RFFT STRUCTURE WITH BUTTERFLY BLOCK SIZE 4

signal	**clock** cycle	**Data-selector-1** **Input-block** (in sample index)	**Output-block** (in sample index)	**remark**
$\mathbf{r}_1$	$CC_{1.1}$	$\{0,16,8,24,4,20,12,28\}$	$\{0,16,8,24,4,20,12,28\}$	no swapping
	$CC_{1.2}$	$\{2,18,10,26,6,22,14,30\}$	$\{2,18,10,26,6,22,14,30\}$	no swapping
	$CC_{1.3}$	$\{1,17,9,24,5,21,13,29\}$	$\{1,17,9,24,5,21,13,29\}$	no swapping
	$CC_{1.4}$	$\{3,19,11,27,7,23,15,31\}$	$\{3,19,11,27,7,23,15,31\}$	no swapping
$\mathbf{u}_1$	$CC_{1.5}$	$\{0,8,16,24,4,12,20,28\}$	$\{0,8,16,24,4,12,20,28\}$	no swapping
	$CC_{1.6}$	$\{2,10,18,26,6,14,2230\}$	$\{2,10,18,26,6,14,2230\}$	no swapping
	$CC_{1.7}$	$\{1,9,17,25,5,13,21,29\}$	$\{1,9,17,25,5,13,21,29\}$	no swapping
	$CC_{1.8}$	$\{3,11,19,27,7,15,23,31\}$	$\{3,11,19,27,7,15,23,31\}$	no swapping
$\mathbf{v}_1$	$CC_{1.9}$	$\{0,4,8,12,16,24,20,28\}$	$\{0,4,8,12,16,24,20,28\}$	no swapping
	$CC_{1.10}$	$\{2,6,10,14,18,26,22,30\}$	$\{10,14,2,6,22,30,18,26\}$	swapping
	$CC_{1.11}$	$\{1,5,9,13,17,25,21,29\}$	$\{1,5,9,13,17,25,21,29\}$	no swapping
	$CC_{1.12}$	$\{3,7,11,15,19,27,23,31\}$	$\{11,15,3,7,23,31,19,27\}$	swapping
$\mathbf{w}_1$	$CC_{1.13}$	$\{0,2,4,6,16,24,18,26\}$	$\{0,2,4,6,16,24,18,26\}$	no swapping
	$CC_{1.14}$	$\{8,12,10,14,20,28,22,30\}$	$\{8,12,10,14,20,28,22,30\}$	no swapping
	$CC_{1.15}$	$\{1,3,5,7,17,25,19,27\}$	$\{5,7,1,3,19,27,17,25\}$	swapping
	$CC_{1.16}$	$\{9,13,11,15,21,29,23,31\}$	$\{11,15,9,13,23,31,21,29\}$	swapping
$\mathbf{x}_2$	$CC_{1.17}$	$\{32,56,48,40,36,60,52,44\}$	$\{32,56,48,40,36,60,52,44\}$	no swapping
	$CC_{1.18}$	$\{34,58,50,42,38,62,54,46\}$	$\{34,58,50,42,38,62,54,46\}$	no swapping
	$CC_{1.19}$	$\{33,57,49,41,37,61,53,45\}$	$\{33,57,49,41,37,61,53,45\}$	no swapping
	$CC_{1.20}$	$\{35,59,51,43,39,63,55,47\}$	$\{35,59,51,43,39,63,55,47\}$	no swapping

TABLE V
INPUT-OUTPUT DATA-FLOW OF DATA-SELECTOR-2 OF PROPOSED RFFT STRUCTURE WITH BUTTERFLY BLOCK SIZE 4

signal	**clock** cycle	**Data-selector-2** **Input-block** (in sample index)	**Output-block** (in sample index)	**remark**
$\mathbf{x}_1$	$CC_{1.1}$	$\{0,24,16,8,4,28,20,12\}$	$\{0,24,16,8,4,28,20,12\}$	no swapping
	$CC_{1.2}$	$\{2,26,18,10,6,30,22,14\}$	$\{2,26,18,10,6,30,22,14\}$	no swapping
	$CC_{1.3}$	$\{1,25,17,9,5,29,21,13\}$	$\{1,25,17,9,5,29,21,13\}$	no swapping
	$CC_{1.4}$	$\{3,27,19,11,7,31,23,15\}$	$\{3,27,19,11,7,31,23,15\}$	no swapping
$\mathbf{r}_1$	$CC_{1.5}$	$\{0,16,8,24,4,20,12,28\}$	$\{0,16,8,24,4,20,12,28\}$	no swapping
	$CC_{1.6}$	$\{2,18,10,26,6,22,14,30\}$	$\{2,18,10,26,6,22,14,30\}$	no swapping
	$CC_{1.7}$	$\{1,17,9,24,5,21,13,29\}$	$\{1,17,9,24,5,21,13,29\}$	no swapping
	$CC_{1.8}$	$\{3,19,11,27,7,23,15,31\}$	$\{3,19,11,27,7,23,15,31\}$	no swapping
$\mathbf{u}_1$	$CC_{1.9}$	$\{0,8,16,24,4,12,20,28\}$	$\{0,8,4,12,16,24,20,28\}$	swapping
	$CC_{1.10}$	$\{2,10,18,26,6,14,2230\}$	$\{2,10,6,14,18,26,2230\}$	swapping
	$CC_{1.11}$	$\{1,9,17,25,5,13,21,29\}$	$\{1,9,5,13,17,25,21,29\}$	swapping
	$CC_{1.12}$	$\{3,11,19,27,7,15,23,31\}$	$\{3,11,7,15,19,27,23,31\}$	swapping
$\mathbf{v}_1$	$CC_{1.13}$	$\{0,4,2,6,16,24,18,26\}$	$\{0,4,2,6,16,24,18,26\}$	no swapping
	$CC_{1.14}$	$\{10,14,8,12,22,30,20,28\}$	$\{8,12,10,14,20,28,22,30\}$	swapping
	$CC_{1.15}$	$\{1,5,3,7,17,25,19,27\}$	$\{1,5,3,7,17,25,19,27\}$	no swapping
	$CC_{1.16}$	$\{11,15,9,13,23,31,21,29\}$	$\{9,13,11,15,21,29,23,31\}$	swapping
$\mathbf{w}_1$	$CC_{1.17}$	$\{0,2,1,3,16,24,17,25\}$	$\{0,2,1,3,16,24,17,25\}$	no swapping
	$CC_{1.18}$	$\{8,12,9,13,20,28,21,29\}$	$\{8,12,9,13,20,28,21,29\}$	no swapping
	$CC_{1.19}$	$\{11,15,10,14,23,31,22,30\}$	$\{10,14,11,15,22,30,23,31\}$	swapping
	$CC_{1.20}$	$\{5,7,4,6,19,27,18,26\}$	$\{4,6,5,7,18,26,19,27\}$	swapping

registers of each bank are accessed through the multiplexer that selects one of the registers as specified by 2-bit read address (r_0, r_1). During each clock cycle, four data values are read from four register-banks of one MCRB such that one block of 8 data are retrieved from MCRB-1 and MCRB-2 to perform the butterfly operation of the i-th BF stage and the intermediate outputs are written back into the same locations (registers) during the next clock transition for the $(i+1)$-th stage of butterfly computations (for $1 \leq i \leq \log_2 N$).

The internal structure of DS-1 and DS-2 is shown in Fig.3. Each input-line ($\mathbf{I}_i$) or output-line ($\mathbf{O}_i$) of DS-1 and DS-2 receives/sends a pair of data samples from/to the BCU, for $(0 \leq i \leq 3)$. The data-swapping operation of DS-1 and DS-2 are controlled by the signal ctr_2. The signal ctr_1 is set to '1' during swapping operation of DS-1. As shown in Fig.2(b), {MUX1, MUX2} swap the data among them for reordering the input-block retrieved from MCRB while {MUX3, MUX4} and {MUX5, MUX6} swap the data to annul the swapping

TABLE VI
OUTPUT DATA-FLOW FROM THE STORAGE-UNIT FOR IN-PLACE RFFT COMPUTATION WITH BUTTERFLY BLOCK SIZE 8 AND FFT SIZE 32

sig	CC	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	B_{15}	B_{16}
		signal sample index															
$\mathbf{x}_1$ stage-1	$CC_{1.1}$	0	24	16	8	4	28	20	12	2	26	18	10	6	30	22	14
	$CC_{1.2}$	1	25	17	9	5	29	21	13	3	27	19	11	7	31	23	15
$\mathbf{r}_1$ stage-2	$CC_{1.3}$	0	16	8	24	4	20	12	28	2	18	10	26	6	22	14	30
	$CC_{1.4}$	1	12	9	25	8	21	13	29	3	19	11	27	7	23	15	31
$\mathbf{u}_1$ stage-3	$CC_{1.5}$	0	8	4	12	16	24	20	28	2	10	6	14	18	26	22	30
	$CC_{1.6}$	1	9	5	13	17	25	21	29	3	11	7	15	19	27	23	31
$\mathbf{v}_1$ stage-4	$CC_{1.7}$	0	4	2	6	16	24	18	26	8	12	10	14	20	28	22	30
	$CC_{1.8}$	1	5	3	7	17	25	19	27	9	13	11	15	21	29	23	31
$\mathbf{w}_1$ stage-5	$CC_{1.9}$	0	2	1	3	16	24	17	25	8	12	9	13	20	28	21	29
	$CC_{1.10}$	4	6	5	7	18	26	19	27	10	14	11	15	22	30	23	31
$\mathbf{x}_2$ stage-1	$CC_{2.1}$	32	56	48	40	36	60	52	44	34	58	50	42	38	62	54	46
	$CC_{2.2}$	33	57	49	41	37	61	53	45	35	59	51	43	39	63	55	47

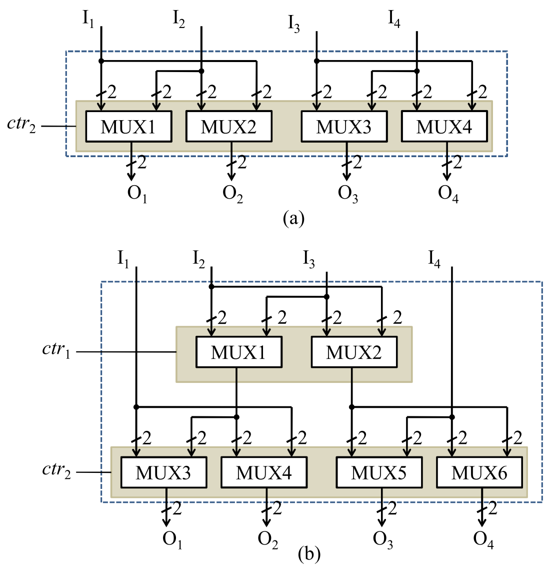

Fig. 3. (a) Data selector-1 (DS-1). (b) Data selector-2 (DS-2).

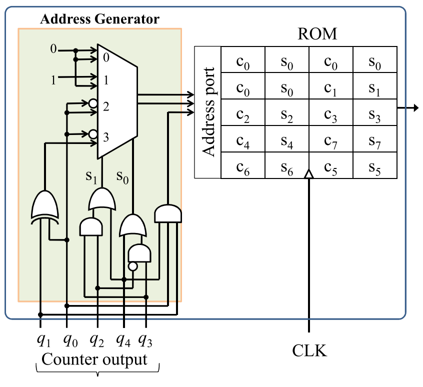

Fig. 4. Structure twiddle factor storage unit for FFT size $N = 32$ and butterfly block-size 4.

operation of DS-1. The design of BCU for butterfly block size 4 is identical to arithmetic unit design of [12] except that it comprises of two complex multipliers and two pairs of line-changers $\{LC_1$ and $LC_2\}$.

The design of twiddle factor storage unit (TSU) for the 32-point RFFT structure and butterfly block-size 4 based on the approach of [12] requires a ROM look-up-table (LUT)

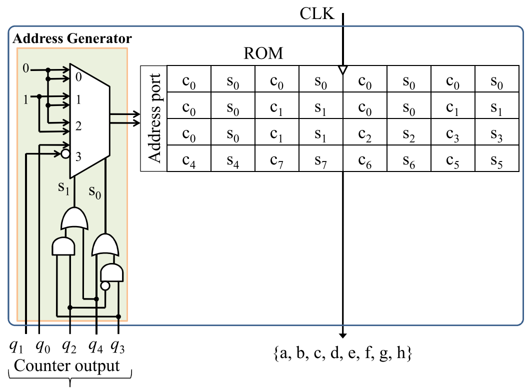

Fig. 5. Structure twiddle factor storage unit for FFT size $N = 32$ and block-size 16.

TABLE VII
DATA SWAPPING OPERATIONS OF DATA-SELECTORS OF RFFT STRUCTURE FOR BUTTERFLY BLOCK-SIZE 8 AND FFT SIZE 32

Data-selector-1			**Data-selector-2**		
Data vector	clock cycle	operation	Data vector	clock cycle	operation
$\mathbf{r}_1$	$CC_{1.1}$	No swapping	$\mathbf{x}_1$	$CC_{1.1}$	No swapping
	$CC_{1.2}$	No swapping		$CC_{1.2}$	No swapping
$\mathbf{u}_1$	$CC_{1.3}$	No swapping	$\mathbf{r}_1$	$CC_{1.3}$	No swapping
	$CC_{1.4}$	No swapping		$CC_{1.4}$	No swapping
$\mathbf{v}_1$	$CC_{1.5}$	No swapping	$\mathbf{u}_1$	$CC_{1.5}$	swapping
	$CC_{1.6}$	No swapping		$CC_{1.6}$	swapping
$\mathbf{w}_1$	$CC_{1.7}$	No swapping	$\mathbf{v}_1$	$CC_{1.7}$	swapping
	$CC_{1.8}$	swapping		$CC_{1.8}$	swapping
$\mathbf{x}_2$	$CC_{1.9}$	No swapping	$\mathbf{w}_1$	$CC_{1.9}$	No swapping
	$CC_{1.10}$	No swapping		$CC_{1.10}$	swapping

of 8-words. However, we find that the TSU stores a few redundant values and that can be avoided to reduce the ROM size. An optimized TSU design for 32-point RFFT with block-size 8 is shown in Fig.4. It involves one ROM LUT of size $(5 \times 4w)$, where w is the wordlength of real twiddle factor constants $\{c_l, s_l\}$. The five BF stages of 32-point FFT are encoded by a 3-bit word $\{q_4, q_3, q_2\}$ to generate the control bits $\{s_1, s_0\}$ of the MUX in order to select the lower 2-bit address of particular BF stage. In general, ROM LUT of size $[(2^{P-3}+1) \times 4w]$ is required for N-point RFFT and block-size

TABLE VIII
COMPARISON OF THEORETICAL ESTIMATE OF HARDWARE AND TIME COMPLEXITIES

Structures	BFB size	Mult	Add	REG/RAM*	MUX/DMUX	ROM (bits)	Minimum cycle period	CT (cc)
Ayinala [10]	4	8	12	$4N$	140	$2^{P-1} \times 2w$	$T_{MA} + T_A + 11T_{MX} + 2T_R$	$(N/8)\log_2 N$
(Ayinala [10]	8	16	24	$4N$	536	$2^{P-1} \times 2w$	$T_{MA} + T_A + 13T_{MX} + 2T_R$	$(N/16)\log_2 N$
Meher [12]	2	4	6	N	$N + 12$	$2^{P-1} \times 2w$	$T_{MA} + T_{FA} + 5T_{MX} + T_{RM}$	$(N/4)\log_2 N$
Proposed	4	8	12	N	$N + 28$	$(2^{P-3} + 1) \times 4w$	$T_{MA} + T_{FA} + 6T_{MX} + T_{RM}$	$(N/8)\log_2 N$
Proposed	8	16	24	N	$N + 64$	$(2^{P-4} + 2) \times 8w$	$T_{MA} + T_{FA} + 6T_{MX} + T_{RM}$	$(N/16)\log_2 N$

*Single-port w-bit RAM, where w is the wordlength. BLB: butterfly block, T_{MA}: computation time of multiplication followed by addition, T_A: real adder delay, T_{FA}: full-adder delay, T_{MX}: 2:1 MUX delay, T_R: RAM access delay, T_{RM}: register-based memory-bank access delay.

8, where $P = \log_2 N$. From Table III, Table IV and Table V, it can be observed that the data loading and retrieval from the data-storage unit of proposed 32-point RFFT structure for block-size 8 is almost identical to those of the existing 16-point RFFT structure of block-size 4 [12]. Therefore, the control unit for generation of read/write addresses of the proposed multi-channel memory banks and control signals for data-selectors, and select signals of AU can be designed by using the scheme given in [12].

IV. STORAGE UNIT DESIGN FOR BLOCK-SIZE 16

The SU of RFFT structure for butterfly block-size 8 is partitioned into 16 memory banks (B_1, B_2, $\cdots$, B_{15}, B_{16}), so that a block of 16 words can be retrieved/stored from/into the 16 banks in each clock cycle using a common address for reference. The output data-flow from the register-banks for computation of different stages of FFT computation of two successive 32-point input vectors $\mathbf{x}_1 = \{x(0), x(1), \cdots, x(31)\}$ and $\mathbf{x}_2 = \{x(32), x(17), \cdots, x(63)\}$ is given in Table VI for 10 clock cycles to study the access conflicts.

As shown in Table VI samples of each input data-block of stage-1, stage-2, stage-3, and stage-4 are read from 16 different banks where no access conflicts are found to occur. However, during each clock cycle of stage-5, the input-blocks are sourced from 8 banks which results in bank conflict. Also it can be observed from Table VI that the sample orders of data-blocks of stage-3 and stage-4 are different from those of stage-2 and stage-3. The data-blocks retrieved from the banks during stage-3 and stage-4 of computation need to be reordered before they are sent to the arithmetic unit. In general, we observe that the access conflict is reduced even for larger block sizes and the data is read from the storage unit. Therefore, DS-1 is expected to perform less swapping while DS-2 need to perform more swapping for higher block sizes. Data-swapping operations of DS-1 and DS-2 are given in Table VII for butterfly block-size 8 and FFT size 32.

V. HARDWARE AND TIME COMPLEXITY

The hardware and time complexity of proposed structure and those of the structures of [10] and [12] are listed in Table VIII for comparison. The structure of [10] uses RAM to implement the data storage unit whereas the proposed structure and the structure of [12] use register-based data-storage. The structure of [10] involves $4N$ single-port RAM words of w-bit width or $2N$ dual-port RAM words of $2w$ bit-width for N-point in-place RFFT computation where the storage unit of proposed structure and the structure of [12] requires N registers. As shown in Table VIII, the proposed structure for butterfly block-size 4 and 8, respectively, offers 2 times and 4 times higher throughput compared to the structure of [12] at the cost of 2 times and 4 times more multipliers and adders, using the same number of registers, less ROM memory than those required by the other. Compared with the structure of [10], the proposed structure for butterfly block-size 8 and 16, respectively, involves the same number of multipliers and adders, N registers against $4N$ RAM words, less ROM and offers higher throughput due to less critical path delay (CPD). Unlike the structure of [10], the CPD of the proposed structure does not change much with increase in butterfly block-size due to reduction in bank-access delay (T_{MR}). In general, the hardware complexity of the proposed structure does not increase proportionately with block-size. Therefore, the area-delay efficiency of proposed structure is better for higher block-sizes.

TABLE IX
SYNTHESIS RESULTS OF PROPOSED STRUCTURES AND EXISTING STRUCTURES USING TMSC 65nm CMOS STANDARD CELL LIBRARY

Design	**FFT size**	**MCP** (ns)	**Area** (um^2)	**Power** (mW)	**ADP** (um^{2}us)	**EPS** (nJ)
Structure of [10] BF=4	32	1.48	25375	9.36	0.75	8.66
	64	1.49	35138	12.52	2.51	13.99
	128	1.52	53023	15.22	9.03	20.24
Structure of [10] BF=8	32	1.56	52541	23.16	0.82	11.29
	64	1.58	59150	24.89	2.24	14.75
	128	1.60	77903	27.59	6.98	19.31
Proposed BF=4	32	1.37	15498	5.67	0.43	4.85
	64	1.42	20087	6.30	1.37	6.71
	128	1.47	32119	5.06	5.29	6.51
Proposed BF=8	32	1.32	23290	12.40	0.31	5.12
	64	1.34	31443	12.63	1.01	6.35
	128	1.37	42410	12.91	3.25	7.74

BF: butterfly block-size, power estimated at MCP for both designs, area delay product (ADP)=area× MCP× CT, energy per sample (EPS)=power× MCP × CT/N, where CT=$(N/L)\log_2 N$.

We have coded the proposed design in VHDL for butterfly block sizes {4 and 8}, and RFFT of length $32, 64$ and 128. We have also coded the structures of [10] for the same FFT sizes and butterfly block size, respectively. We have used single-port RAM generated by Synopsis DesignWare for the implementation of storage unit of [10] whereas delay flip-flop (D-FF) is used for the implementation of register-based data storage unit of proposed design. We have taken 8-bit word-length for the input and the output for all stages of computation. We have synthesized all the designs using TSMC 65nm CMOS standard cell library. The area, minimum cycle period (MCP), and power consumption reported by Synopsys Design Compiler are listed in Table IX. Power consumption is estimated at the MCP of respective designs. As shown in Table IX, for butterfly block-size 4 and 8, the proposed structure involves nearly (5% and 14%) less MCP, (40% and 49%) less area, and (52% and 50%) less power than those of [10] on average for different FFT lengths, respectively. For the same butterfly block-sizes, the proposed structure offers ($\sim$ 44% and $\sim$ 57%) ADP[3] saving and ($\sim$ 54% and $\sim$ 57%) EPS[4] saving than those of [10] on average for different FFT lengths, respectively.

VI. Conclusions

We have proposed a design approach to develop efficient architecture for in-place RFFT which could be scaled for higher throughput and larger FFT sizes. The proposed structure for butterfly block size 4 and 8, involves significantly less EPS than the existing designs. The proposed design for butterfly block size 16 can meet the throughput requirement of Ultra wide band (UWB) and WSN applications while the proposed design for butterfly block sizes 8 can able to meet the throughput requirement of WSN and medical implantable devices.

References

[1] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," *IEEE Trans. Comput.*, vol. C-33, no. 5, pp. 414–426, May 1984.

[2] Y.-N. Chang and K. K. Parhi, "An efficient pipelined FFT architecture," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 50, no. 6, pp. 322–325, Jun. 2003.

[3] H. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, no. 6, pp. 849–863, Jun. 1987.

[4] S.-F. Hsiao and W.-R. Shiue, "Design of low-cost and high-throughput linear arrays for DFT computations: Algorithms, architectures, and implementations," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 11, pp. 1188–1203, Nov. 2000.

[5] M. Garrido, K. K. Parhi, and J. Grajal, "A pipelined FFT architecture for real-valued signals," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 12, pp. 2634–2643, Dec. 2009.

[6] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1068–1081, Jun. 2012.

[7] A. Wang and A. P. Chandrakasan, "Energy-aware architectures for a real-valued FFT implementation," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 2003, pp. 360–365.

[8] H.-F. Chi and Z.-H. Lai, "A cost-effective memory-based real-valued FFT and Hermitian symmetric IFFT processor for DMT-based wire-line transmission systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 6, May 2005, pp. 6006–6009.

[9] L. G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 5, pp. 312–316, May 1992.

[10] M. Ayinala, Y. Lao, and K. K. Parhi, "An in-place FFT architecture for real-valued signals," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 10, pp. 652–656, Oct. 2013.

[3]ADP=area$\times$ MCP$\times(N/L)\log_2 N$
[4]EPS=power$\times$ MCP$\times[(N/L)\log_2 N]/N$

[11] B. G. Jo and M. H. Sunwoo, "New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 911–919, May 2005.

[12] P. K. Meher, B. K. Mohanty, S. K. Patel, S. Ganguly, and T. Srikanthan, "Efficient VLSI architecture for decimation-in-time fast Fourier transform of real-valued data," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 12, pp. 2836–2845, Dec. 2015.

[13] S. R. Kuang, J. P. Wang, and C. Y. Guo, "Modified Booth multipliers with a regular partial product array," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 5, pp. 404–408, May 2009.

Basant K. Mohanty (M'06–SM'11) received the M.Sc. degree in physics from Sambalpur University in 1989 and the Ph.D. degree in the field of VLSI for digital signal processing from Berhampur University, Orissa, in 2000. He joined the EEE Department, BITS Pilani, Rajasthan, in 2001, ECE Department, Mody Institute of Education Research, Rajasthan, in 2002, and the Jaypee University of Engineering and Technology, Guna, in 2003, where he becomes a Full Professor in 2007. He is currently associated with SVKM's NMIMS, Mukesh Patel School of Technology Management and Engineering Shirpur Campus, as a Professor with the Department of Electronics and Telecommunication Engineering. His research interests include the design and implementation of high-performance digital systems for signal processing applications, approximate computation and reconfigurable architecture. He has published nearly 65 technical papers, which include 17 papers in IEEE Transactions. He is currently serving as an Associate Editor for the *Circuits, Systems, and Signal Processing*. He is a Life Time Member of the Institution of Electronics and Telecommunication Engineering, New Delhi, India, and he was a recipient of the Rashtriya Gaurav Award conferred by the India International Friendship Society, New Delhi, in 2012.

Pramod Kumar Meher (SM'03) is currently a Hardware Design Consultant and with the C. V. Raman College of Engineering, Bhubaneswar, India, as Research Advisor. Previously, he has served in senior research positions with Nanyang Technological University, Singapore, and the Institute for Infocomm Research, Singapore, from 2004 to 2016. His research interests include the design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal, image and video processing, communication, bio-informatics, and intelligent computing. He has contributed over 250 technical papers to various reputed journals and conference proceedings, which includes over 80 papers in IEEE Journals.

Dr. Meher has served as a Speaker for the Distinguished Lecturer Program of the IEEE Circuits Systems Society from 2011 to 2012. He has also served as an Associate Editor for the IEEE Transactions on Circuits and Systems for Video Technology, the IEEE Transactions on Very Large Scale Integration Systems, the IEEE Transactions on Circuits and Systems-I: Regular Papers, and the IEEE Transactions on Circuits and Systems-II: Express Briefs during the last 10 years. He is currently an Associate Editor of the IEEE Transactions on Circuits and Systems-I and the *Journal of Circuits, Systems, and Signal Processing*. He is a Fellow of the Institution of Electronics and Telecommunication Engineers, India. He was a recipient of the Samanta Chandrasekhar Award for excellence in research in engineering and technology for 1999.